WIKIPEDIA
The Free Encyclopedia

Filename extension

A **filename extension**, **file name extension** or **file extension** is a suffix to the name of a computer file (for example, `.txt`, `.mp3`, `.exe`) that indicates a characteristic of the file contents or its intended use. A filename extension is typically delimited from the rest of the filename with a full stop (period), but in some systems[1] it is separated with spaces.

Some file systems, such as the FAT file system used in DOS, implement filename extensions as a feature of the file system itself and may limit the length and format of the extension, while others, such as Unix file systems, the VFAT file system, and NTFS, treat filename extensions as part of the filename without special distinction.

Operating system and file system support

The Multics file system stores the file name as a single string, not split into base name and extension components, allowing the "." to be just another character allowed in file names. It allows for variable-length filenames, permitting more than one dot, and hence multiple suffixes, as well as no dot, and hence no suffix. Some components of Multics, and applications running on it, use suffixes to indicate file types, but not all files are required to have a suffix — for example, executables and ordinary text files usually have no suffixes in their names.

File systems for UNIX-like operating systems also store the file name as a single string, with "." as just another character in the file name. A file with more than one suffix is sometimes said to have more than one extension, although terminology varies in this regard, and most authors define *extension* in a way that does not allow more than one in the same file name. More than one extension usually represents nested transformations, such as `files.tar.gz` (the `.tar` indicates that the file is a tar archive of one or more files, and the `.gz` indicates that the tar archive file is compressed with gzip). Programs transforming or creating files may add the appropriate extension to names inferred from input file names (unless explicitly given an output file name), but programs reading files usually ignore the information; it is mostly intended for the human user. It is more common, especially in binary files, for the file to contain internal or external metadata describing its contents. This model generally requires the full filename to be provided in commands, whereas the metadata approach often allows the extension to be omitted.

CTSS was an early operating system in which the filename and file type were separately stored. Continuing this practice, and also using a dot as a separator for display and input purposes (while not storing the dot), were various DEC operating systems (such as RT-11), followed by CP/M and subsequently DOS.

In DOS and 16-bit Windows, file names have a maximum of 8 characters, a period, and an extension of up to three letters. The FAT file system for DOS and Windows stores file names as an 8-character name and a three-character extension. The period character is not stored.

The High Performance File System (HPFS), used in Microsoft and IBM's OS/2 stores the file name as a single string, with the "." character as just another character in the file name. The convention of using suffixes continued, even though HPFS supports extended attributes for files, allowing a file's type to be stored in the file as an extended attribute.

Microsoft's Windows NT's native file system, NTFS, and the later ReFS, also store the file name as a single string; again, the convention of using suffixes to simulate extensions continued, for compatibility with existing versions of Windows. In Windows NT 3.5, a variant of the FAT file system, called VFAT appeared; it supports longer file names, with the file name being treated as a single string.

Windows 95, with VFAT, introduced support for long file names, and removed the 8.3 name/extension split in file names from non-NT Windows.

The classic Mac OS disposed of filename-based extension metadata entirely; it used, instead, a distinct file type code to identify the file format. Additionally, a creator code was specified to determine which application would be launched when the file's icon was double-clicked.[2] macOS, however, uses filename suffixes as a consequence of being derived from the UNIX-like NeXTSTEP operating system, in addition to using type and creator codes.

In Commodore systems, files can only have four extensions: PRG, SEQ, USR, REL. However, these are used to separate data types used by a program and are irrelevant for identifying their contents.

With the advent of graphical user interfaces, the issue of file management and interface behavior arose. Microsoft Windows allowed multiple applications to be associated with a given extension, and different actions were available for selecting the required application, such as a context menu offering a choice between viewing, editing or printing the file. The assumption was still that any extension represented a single file type; there was an unambiguous mapping between extension and icon.

When the Internet age first arrived, those using Windows systems that were still restricted to 8.3 filename formats had to create web pages with names ending in `.HTM`, while those using Macintosh or UNIX computers could use the recommended `.html` filename extension. This also became a problem for programmers experimenting with the Java programming language, since it *requires* the four-letter suffix `.java` for source code files and the five-letter suffix `.class` for Java compiler object code output files.[3]

Content type

Filename extensions may be considered a type of metadata.[4] They are commonly used to imply information about the way data might be stored in the file. The exact definition, giving the criteria for deciding what part of the file name is its extension, belongs to the rules of the specific file system used; usually the extension is the substring which follows the last occurrence, if any, of the dot character (*example:* `txt` is the extension of the filename `readme.txt`, and `html` the extension of `index.html`). On file systems of some mainframe systems such as CMS in VM, VMS, and of PC systems such as CP/M and derivative systems such as MS-DOS, the extension is a separate namespace from the filename. Under Microsoft's DOS and Windows, extensions such as `EXE`, `COM` or `BAT` indicate

that a file is a program executable. In OS/360 and successors, the part of the dataset name following the last period, called the low level qualifier, is treated as an extension by some software, e.g., TSO EDIT, but it has no special significance to the operating system itself; the same applies to Unix files in MVS.

The filename extension was originally used to determine the file's generic type. The need to condense a file's type into three characters frequently led to abbreviated extensions. Examples include using `.GFX` for graphics files, `.TXT` for plain text, and `.MUS` for music. However, because many different software programs have been made that all handle these data types (and others) in a variety of ways, filename extensions started to become closely associated with certain products—even specific product versions. For example, early WordStar files used `.WS` or `.WS`*n*, where *n* was the program's version number. Also, conflicting uses of some filename extensions developed. One example is `.rpm`, used for both RPM Package Manager packages and RealPlayer Media files;.[5] Others are `.qif`, shared by DESQview fonts, Quicken financial ledgers, and QuickTime pictures;[6] `.gba`, shared by GrabIt scripts and Game Boy Advance ROM images;[7] `.sb`, used for SmallBasic and Scratch; and `.dts`, being used for Dynamix Three Space and DTS.

Compared to MIME type

In many Internet protocols, such as HTTP and MIME email, the type of a bitstream is stated as the media type, or MIME type, of the stream, rather than a filename extension. This is given in a line of text preceding the stream, such as *Content-type: text/plain*.

There is no standard mapping between filename extensions and media types, resulting in possible mismatches in interpretation between authors, web servers, and client software when transferring files over the Internet. For instance, a content author may specify the extension *svgz* for a compressed Scalable Vector Graphics file, but a web server that does not recognize this extension may not send the proper content type *application/svg+xml* and its required compression header, leaving web browsers unable to correctly interpret and display the image.

BeOS, whose BFS file system supports extended attributes, would tag a file with its media type as an extended attribute. Some desktop environments, such as KDE Plasma and GNOME, associate a media type with a file by examining both the filename suffix and the contents of the file, in the fashion of the file command, as a heuristic. They choose the application to launch when a file is opened based on that media type, reducing the dependency on filename extensions. macOS uses both filename extensions and media types, as well as file type codes, to select a Uniform Type Identifier by which to identify the file type internally.

Executable programs

The use of a filename extension in a command name appears occasionally, usually as a side effect of the command having been implemented as a script, e.g., for the Bourne shell or for Python, and the interpreter name being suffixed to the command name, a practice common on systems that rely on associations between filename extension and interpreter, but sharply deprecated[8] in Unix-like systems, such as Linux, Oracle Solaris, BSD-based systems, and Apple's macOS, where the interpreter is normally specified as a header in the script ("shebang").

On association-based systems, the filename extension is generally mapped to a single, system-wide selection of interpreter for that extension (such as ".py" meaning to use Python), and the command itself is runnable from the command line even if the extension is omitted (assuming appropriate setup is done). If the implementation language is changed, the command name extension is changed as well, and the OS provides a consistent API by allowing the same extensionless version of the command to be used in both cases. This method suffers somewhat from the essentially global nature of the association mapping, as well as from developers' incomplete avoidance of extensions when calling programs, and that developers can not force that avoidance. Windows is the only remaining widespread employer of this mechanism.

On systems with interpreter directives, including virtually all versions of Unix, command name extensions have no special significance, and are by standard practice not used, since the primary method to set interpreters for scripts is to start them with a single line specifying the interpreter to use. In these environments, including the extension in a command name unnecessarily exposes an implementation detail which puts all references to the commands from other programs at future risk if the implementation changes. For example, it would be perfectly normal for a shell script to be reimplemented in Python or Ruby, and later in C or C++, all of which would change the name of the command were extensions used. Without extensions, a program always has the same extension-less name, with only the interpreter directive or magic number changing, and references to the program from other programs remain valid.

Security issues

File extensions alone are not a reliable indicator of a file's type, as the extension can be modified without changing the file's contents, such as to disguise malicious content. Therefore, especially in the context of cybersecurity, a file's true nature should be examined for its signature, which is a distinctive sequence of bytes affixed to a file's header. This is accomplished using file identification software or a hex editor, which provides a hex dump of a file's contents.[9] For example, on UNIX-like systems, it is not uncommon to find files with no extensions at all,[10] as commands such as `file` are meant to be used instead, and will read the file's header to determine its content.

Malware such as Trojan horses typically takes the form of an executable, but any file type that performs input/output operations may contain malicious code. A few data file types such as PDFs have been found to be vulnerable to exploits that cause buffer overflows.[11] There have been instances of malware crafted to exploit such vulnerabilities in some Windows applications when opening a file with an overly long, unhandled filename extension.

File managers may have an option to hide filenames extensions. This is the case for File Explorer, the file browser provided with Microsoft Windows, which by default does not display extensions. Malicious users have tried to spread computer viruses and computer worms by using file names formed like `LOVE-LETTER-FOR-YOU.TXT.vbs`. The idea is that this will appear as `LOVE-LETTER-FOR-YOU.TXT`, a harmless text file, without alerting the user to the fact that it is a harmful computer program, in this case, written in VBScript.[11] The default behavior for ReactOS is to display filename extensions in ReactOS Explorer. Later Windows versions (starting with Windows XP Service Pack 2 and Windows Server 2003) included customizable lists of filename extensions that should be considered "dangerous" in certain "zones" of operation, such as when downloaded from the web or received as an e-mail attachment. Modern antivirus software systems also help to defend users against such attempted attacks where possible.

A virus may couple itself with an executable without actually modifying the executable. These viruses, known as *companion viruses*, attach themselves in such a way that they are executed when the original file is requested. One way such a virus does this involves giving the virus the same name as the target file, but with a different extension to which the operating system gives priority, and often assigning the former a "hidden" attribute to conceal the malware's existence. The efficacy of this approach depends on whether the user attempts to open the intended file by entering a command and whether the user includes the extension. Later versions of DOS and Windows check for and attempt to run `.COM` files first by default, followed by `.EXE` and finally `.BAT` files. In this case, the infected file is the one with the `.COM` extension, which the user unwittingly executes.[10][11]

Some viruses take advantage of the similarity between the ".com" top-level domain and the `.COM` filename extension by emailing malicious, executable command-file attachments under names superficially similar to URLs (*e.g.*, "myparty.yahoo.com"), with the effect that unaware users click on email-embedded links that they think lead to websites but actually download and execute the malicious attachments.

See also

- file (command)
- List of file formats
- List of filename extensions
- Metadata
- .properties

References

1. "What Is a File?" (https://www.ibm.com/servers/resourcelink/svc0302a.nsf/pages/zVMV7R2sc246265/$file/dmsb2_v7r2.pdf#page=23) (PDF). *z/VM 7.2 CMS Primer* (https://www.ibm.com/servers/resourcelink/svc0302a.nsf/pages/zVMV7R2sc246265/$file/dmsb2_v7r2.pdf) (PDF). IBM. 2021-12-05. p. 7. SC24-6265-01. "One thing you need to know about creating files with z/VM is that each file needs its own three-part identifier. The first part of the identifier is the file name. The second part is the file type. And the third part is the file mode. These three file identifiers are often abbreviated fn ft fm."
2. "Mac Creator and File Type codes" (https://livecode.byu.edu/helps/file-creatorcodes.php). *livecode.byu.edu*. Retrieved 2022-09-02.
3. "javac – Java programming language compiler" (http://java.sun.com/j2se/1.5.0/docs/tooldocs/windows/javac.html). Sun Microsystems, Inc. 2004. Retrieved 2009-05-31. "Source code file names must have .java suffixes, class file names must have .class suffixes, and both source and class files must have root names that identify the class."
4. Stauffer, Todd; McElhearn, Kirk (2006). *Mastering Mac OS X* (https://books.google.com/books?id=62xkJo6JXwAC&pg=PA95). John Wiley & Sons. pp. 95–96. ISBN 9780782151282. Retrieved 2 October 2017.
5. File Extension .RPM Details (http://filext.com/file-extension/rpm) from filext.com
6. File Extension .QIF Details (http://filext.com/file-extension/qif) from filext.com
7. File Extension .GBA Details (http://filext.com/file-extension/GBA) from filext.com
8. Commandname Extensions Considered Harmful (http://www.talisman.org/~erlkonig/documents/commandname-extensions-considered-harmful)

9. Aquilina, James M.; Casey, Eoghan; Malin, Cameron H. (2008). *Malware Forensics: Investigating and Analyzing Malicious Code* (https://books.google.com/books?id=lRjO8opcPzIC&q=extension%20signature&pg=PA211). Syngress. pp. 211, 298–299. ISBN 978-1-59749-268-3. Retrieved 2025-02-25.
10. Skoudis, Ed; Zeltser, Lenny (2004). *Malware: Fighting Malicious Code* (https://books.google.com/books?id=TKEAQmQV7O4C&pg=PA32). Prentice Hall. pp. 32–34, 253–254. ISBN 0-13-101405-6. Retrieved 2025-02-25.
11. Grimes, Roger (August 2001). *Malicious Mobile Code: Virus Protection for Windows* (https://books.google.com/books?id=1HYlDwAAQBAJ&pg=PA71). O'Reilly Media. pp. 41–42, 71–74, 221–222, 395–396, 422. ISBN 1-56592-682-X. Retrieved 2025-02-25.

External links

- Media related to Filename extensions at Wikimedia Commons
- Database of filename extensions (https://fileinfo.com/) at FileInfo.com

Retrieved from "https://en.wikipedia.org/w/index.php?title=Filename_extension&oldid=1292033826"